

DESIGN OF OPEN ARCHITECTURE SHIP ALARM, MONITORING AND CONTROL SYSTEM

M. S. ZAGHLOUL

Arab Academy for Science and Technology and Maritime Transport, Department of Electronic and Communication,
Alexandria, Egypt

ABSTRACT

This paper is practical design and implementation of open architecture control, alarm and monitoring system. Modern ships have an automatic system control which includes control, alarm and monitoring system that have access to all process control station and can monitor them. The control system control several types of self-running process control station, each type is dictated to specific task. The alarm system is connected to sensors everywhere in the ship and continuously monitors them, if any sensor reading is outside the preset limits we get an alarm. The monitoring system can record any alarm status and save it in hard disk or printer with time stamp.

The alarm system depends mainly on data coming from different sensors connected to corresponding measuring points. In small ship this system can be small LCD and LED panel but in a large ship it can be a computer and many displays. The sensor data can be current loop, voltage contact or open collector digital data. The number of points and scan rate is considered, also an inhibit control can be applied to certain alarm group for disable at certain conditions for the system.

KEYWORDS: IMO, SOLAS, Ship Control, Signal Processing, Sensors

INTRODUCTION

Within the environment of a marine plant there are many parameters which need to be controlled or monitored including: temperatures, pressure, level, viscosity, flow control, speed, torque control, voltage, current, machinery status (on/ off), and equipment status (open/ closed). In olden times it was the role of the watch keeping engineers to monitor and control the machinery plant. This was achieved by periodically taking rounds around the engine room and manually inspecting the condition of the running machinery. Often the engineer was totally dependent on his natural senses, frequently supported by only the minimum of widely distributed simple monitoring devices. [1] This is an example design of an open architecture system.

The intention here is to use microcontrollers without cooling to avoid using rugged or marine computers which are expensive. To be able to use different number of boards and different types of boards a Data Bus for the microcontrollers is built such that it can accept general type boards and this data bus can accept hundreds of boards.

As an example of board types are- Digital Input board (Isolated), Digital output board (Isolated), Analog input board- Analog output board and Relay output boards.[2] We had to use a simple microcontroller (Microchip PIC16F877) which is available in market. We had to use processing board with more than one microcontroller (Multi controller

technique) with a communication protocol between them to avoid crash. We will go through the design in this sequence: Processing board and data bus, Digital then Analog Input board, Panel of LEDs Program.

PROCESSING BOARD AND DATA BUS

Processing Board

The main board contain 2 microcontrollers PIC16F877 working separately but both of them can access SEREEPROM according to a protocol. Figure 1 shows a block diagram for the design. As shown Microcontroller 1 (MCU1) is interfaced to the following parts:

LCD which is a 4 Lines module each line can have 20 characters. This module has a parallel interface with port C of MCU1. We used 4 bits from port C (C4, C5, C6, and C7) only.

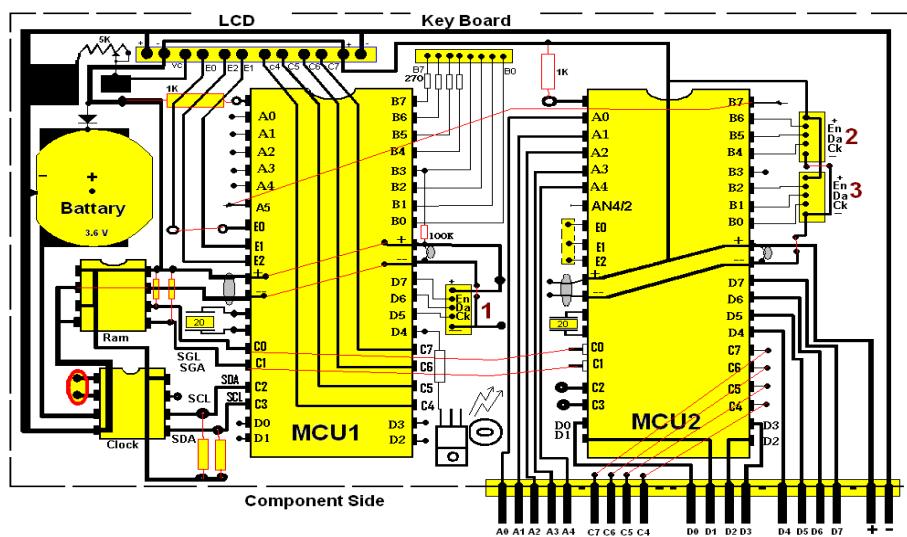


Figure 1: Interconnection Diagram for the Design

The LCD is controlled by (E0, E1, and E2) from MCU1, Serial Data Interface to connect MCU1 with any number of LEDs. This connection is established with (D5, D6, and D7), Key Board which is 4x4 buttons in matrix arrangement. They are connected to PORT B at MCU1, RAM connection which is a serial interface from (C0, C1) and Clock module (RTC) is connected to MCU1 at C2; C3 this module uses a backup battery.6-Buzzer connected at D4 of MCU1. We have also Microcontroller 2 (MCU2) which is interfaced to the following parts: Data Bus which uses PORT D (8 bits),, RAM connection which is a serial interface from (C0, C1) and Quantity 2 Serial Data Interfaces to connect MCU2 with any number of external devices these interfaces use PORT B of MCU2. The control unit in this system has a key board for entering the orders or data [3]. It contains also the power switch and an LCD display. Figure 2 shows a general view of the control unit.



Figure 2: General View of the Control Unit

Internally this unit contains the following items: Main processing board, Key board, LCD display, and Power switch. Figure 3 shows a general view of the main processing board connected to its interfaces.

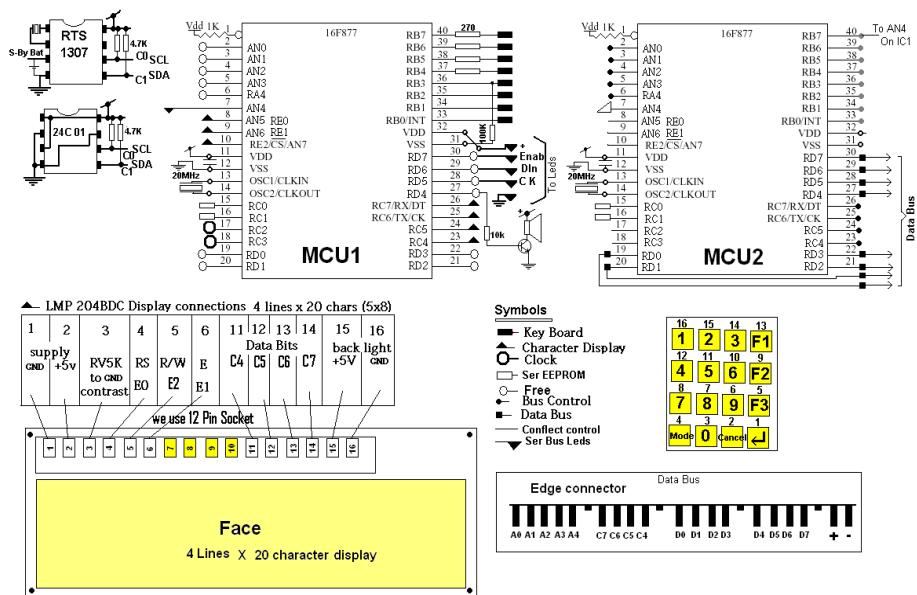


Figure 3: General View of the Main Processing Board with its Interfaces

This board contains the following parts: 2 MCUI units (MCU1 & MCU2), Ser EEPROM, Real time clock and Battery, Buzzer and driving transistor, and Edge connectors for LCD, Key board, Led units and the data bus. The key board is a 4 X4 matrix Keyboard. The LCD Display is a Character LCD display which has 4 lines of characters each line is 20 characters long.

Data Bus

This Bus comes from MCU2 which include 8 data Bits (D0....D7), and some other control lines (A0....A4) & (C4...C7), It includes also the supply lines for 5 volts. This data Bus has a termination resistors (5K pulled to +V) and is likely places near the last boards in the Bus. Likely places near the last boards in the bus.

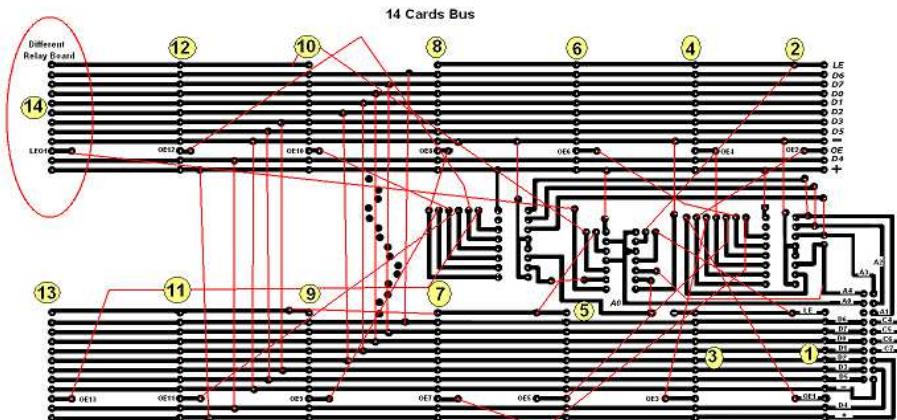


Figure 4: General Data Bus Connection in Detail

Our data bus is theoretically, not limited and it can be tailored to any system according to the number of boards required. In this demo system we have slots 1, 2 for digital input boards Slot 7 for analog input board.

DIGITAL AND ANALOG INPUT BOARD

Digital Input Board

The digital input board receives and pass to the data bus the status of a contact (Relay or Switch). It can be normally closed or normally open contact.

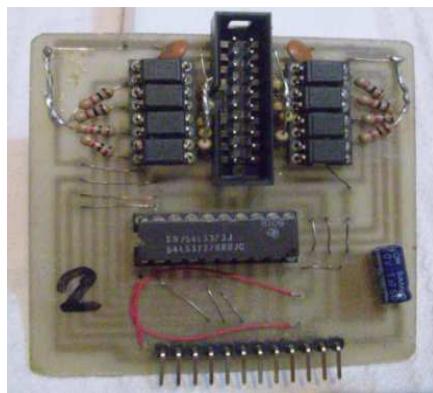


Figure 5: Digital Input Board

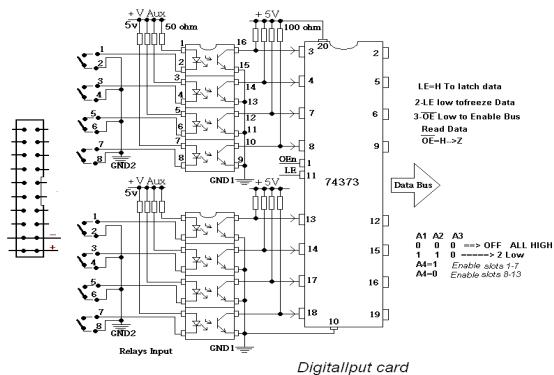


Figure 6: Diagram of Digital Input Board

We use opt isolators at the receiving terminals to completely isolate the system from any monitor device. This small board can accept 8 inputs.

Analog Input Board

The analog input board receives and passes to the data bus the measured value of mill amperes coming from analog sensors (Current loop). It can be 4-20 m Amp this small board.

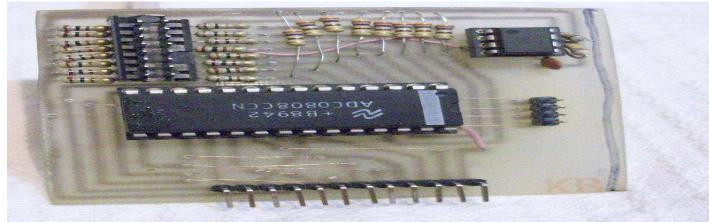


Figure 7: Analog Input Board

The analog input board receives and passes to the data bus the measured value of mill amperes coming from analog sensors (Current loop). It can be 4-20 m Amp. This small board can accept 8 analog inputs the ADC used have 8 bit sampling accuracy. Figure 8 shows the Idea of measuring the mill ampere. By connecting this input circuit in series with the measured mill amperes. The current passing in the resistor 0 to 250 Ohm produce voltage drop value from 4-20 m Amp leads to 1 -5Volt.

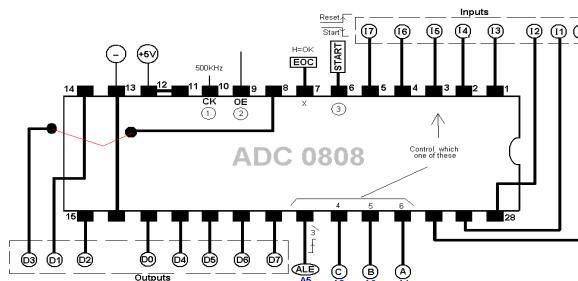


Figure 8: Analog Input Board Connections

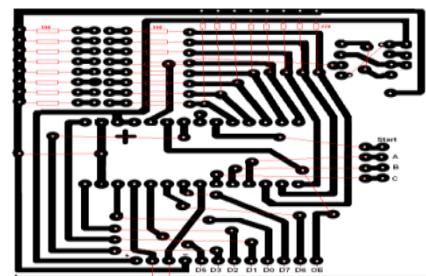


Figure 9: Analog Input Board Wire Connections

INDICATORS FOR SHIP ALARM AND CONTROL SYSTEM

Panel of LEDs

This Panel is the quick indication unit.

When there is a fault a buzzer is heard and after 5 seconds we see the corresponding led becoming red. Also we see a written message on the LCD display showing the name of the faulty part. Figure 10 shows an image of the Led panel.

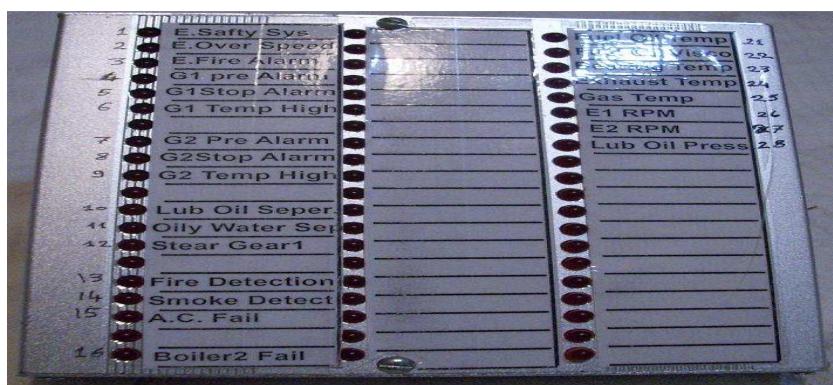


Figure 10: The Connection Diagram for the LEDs

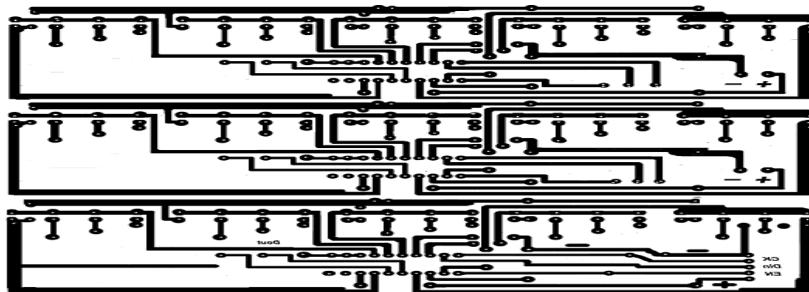


Figure 11: Circuit Uses a Serial Interface with the 1st Microcontroller (MCU1)

The Used Program

We mean by the program the software inside MCU1 which is shown in Appendix A, the software inside MCU2 is shown in Appendix B. [both of them is available under request from the author]

CONCLUSIONS

The system is inexpensive and versatile for testing ships by using alarm and monitoring control. This control system use for ships which has both digital and analog circuit boards interface. The proposed designed control system control several types of self-running process control station, each type is dictated to specific task. The alarm system is connected to sensors everywhere in the ship and continuously monitors them, if any sensor reading is outside the preset limits we get an alarm. The monitoring system can record any alarm status and save it in hard disk or printer with time stamp. Adding to that it can be used as base to test variety of all ship status from the bridge. The implementation of this control system increases the dependency, reliability, reduces the manpower and reduces the cost; also it can be used to add features and fix design bugs after fabrication of a complete ship system moreover reprogram ability yields to a much more engineer friendly system.

The demand to reduce manning level led to the development of automatic control arrangements for the engine room plant which enabled unattended operation of machinery **spaces** [5]. With vessels capable of safe operation for any period of time in this mode, all the control systems and monitoring facilities are grouped together in an Engine Control Room for example. The system consume little power in order of only a few milliamperes per interface and driver is protected such that it often prevents the user from destroying sensor that was installed or programmed improperly. This system feature improves the setup and reduces the fall down time [6,7, 8]. The straightforward programming design interface allows the multi use of the design in different cases specially when installing or upgrading sensor or system in the ship. Applying this system high speed for testing is achieved and can be used to test more complex designs according the requirements.

REFERENCES

1. Mohamed H. El-Mahlawy, and Ahamed Seddik, Design and Implementation of New Automatic Testing System for Digital Circuits Based on the Signature Analysis, 12th International Conference on Aerospace Sciences & Aviation Technology, Military Technical College, Egypt, 29-31 May 2007.
2. Sherif Anas Mohamed, in-Circuit Testing for Electronic Board, M.Sc. degree, MTC, Cairo, 2006.

3. Jutman, R. Ubar, V. Hahanov, O. Skvortsova, "Practical Works for on-Line Teaching Design and Test of Digital Circuits," in Proc. of 9th IEEE International Conference on Electronics, Circuits and Systems (ICECS'2002), Dubrovnik, Croatia, Sept. 15-18, 2002, Vol. 3, pp. 1223-1226.
4. <http://www.travelpod.com/photos/8/Italy/Genoa.html>.
5. <http://www.boatnerd.com/pictures/fleet/jamesrbarker.htm>.
6. Mindell, David (2002), between Human and Machine. Baltimore: Johns Hopkins. pp. 20-21, ISBN 0-8018-8057-2.
7. Annals of the History of Computing, Volume 4, Number 3, July 1982 Electrical Computers for Fire Control, p232, W. H. C. Higgins, B. D. Holbrook
8. DiGiulian, Tony (November 2006). "United States of America 40 mm/ 56 mm (1.57") Mark 1, Mark 2 and M1". Navweaps.com. Retrieved 2007-02-25.

APPENDICES

Appendix A

```
'PicBasic Pro program to MCU1Ver1-9.

Define LOADER_USED 1

include "modedefs.bas"

Define OSC 20

Define LCD_DREG  PORTC  '(C4,C5,C6,C7)

Define LCD_DBIT    4

Define LCD_RSREG  PORTE

Define LCD_RSBIT   0

Define LCD_EBIT    1

Define LCD_EREG    PORTE

Pattern1 var byte: Pattern2 var byte

Soudvar PORTD.4

DOUT var PORTD.6  ' Shift data pin

COUT var PORTD.5  ' Shift clock pin

denabvar PORTD.7  'enable the board

iivar  byte: ii=0

SCL  var  PORTC.2  ' Clock pin RTC Clock
```

```

SDA  var  PORTC.3  ' Data pin  RTC Clock
SGL  var  PORTC.0  ' Clock pin for ser EEPROM
SGA  var  PORTC.1  ' Data pin forser EEPROM
B0  var  byte  ' Address
Modeevar  byte  'mode for processor no2.
MModeevar  byte  'Data 2
Con1 VAR byte :Con2 VAR byte :Con3 VAR byte:Con4 VAR byte
Con5 VAR byte:Con6 VAR byte:Con7 VAR byte :Con8 VAR byte
Con9 VAR byte:Con10 VAR byte:Con11 VAR byte:Con12 VAR byte
Con13 VAR byte:Con14 VAR byte:Con15 VAR byte :Con16 VAR byte
Contvarbit[32]
BUSSS var byte[8]
BOSSS var byte[8]
DB0  var byte[8]
colvar  byte  ' Keypad column
rowvar  byte  ' Keypad row
keyvar  byte  ' Key value
Beyvar  byte
keyNamevar Word
h1var byte :h2 var byte :h var byte
s1var byte :s2 var byte :s var byte
m1var byte :m2 var byte :m var byte
ADCON1 = 7  ' Set PORTA and PORTE to digital
OPTION_REG.7 = 0  ' Enable PORTB pullups
TRISB = 0  'OUTPUT:
TRISE.2 = 0
zako:
lowSoud
Low PORTE.2  ' LCD R/W line low (W)

```

```
Pause 100  ' Wait for LCD to start up

Lcdout $fe, 1  ' Clear screen

Pause 100  ' Wait. 5 second

Lcdout" --Arab Academy--"

pause 2000

' Set time & date to 1:20:00 THURUSDAY 26th of AUG 2012

' I2CWRITE SDA,SCL,$D0,$00,[\$00,\$20,\$1,\$4,\$26,\$4,\$12,\$90]

' pause 100

read_1307:

I2CREAD SDA,SCL,$D1,$00,[STR DB0\8]

pause 50

gosubCharr

pause 50

Lcdout $fe, \$c0," Time", hex2 DB0[2],":",hex2 DB0[1],":",hex2 DB0[0]

Pause 5000  ' Wait. 5 second

Lcdout $fe, \$D4, "Press F1 for Options"

Pause 1000

Gosubgetkey

if (key == 13) Then

GosubOptioon

else

Lcdout $fe, 1, " No Options !!""

Pause 2000

endif

pause 1000

Lcdout $fe, 1, "To Set date/Time->F1"  '13

Lcdout $fe, \$c0,"To check system-->F2"  '9

Lcdout $fe, \$94,"To Read Points--->F3"  '5

Lcdout $fe, \$D4,"To RUN System->Enter"  '1
```

```

Pause 1000

Gosubgetkey

if (Key==16) Then  'enter

Gosub cont1

pause 10000

gotozako

endif

if (Key==13) Then  'F1

GosubAdj

gotozako

endif

if (Key==14) Then  'F2

Pattern1= %10101010: Pattern2= %01010101

Gosub Loop0

gotozako

endif

' if (Key==15) Then  'F3

'endif

pause 500

Lcdout $fe, 1, " Restarting >>>"

pause 5000

gotozako

'=====

Charrr:

if DB0[3]==1 then LCDOUT$fe,1,"Monday:",hex2 DB0[4],"/",hex2 DB0[5],"/",hex2 DB0[6]

if DB0[3]==2 then LCDOUT$fe,1,"Tuesday:",hex2 DB0[4],"/",hex2 DB0[5],"20",hex2 DB0[6]

if DB0[3]==3 then LCDOUT$fe,1,"Widnesday:",hex2 DB0[4],"/",hex2 DB0[5],"20",hex2 DB0[6]

if DB0[3]==4 then LCDOUT$fe,1,"Thurusday:",hex2 DB0[4],"/",hex2 DB0[5],"20",hex2 DB0[6]

if DB0[3]==5 then LCDOUT$fe,1,"Friday:",hex2 DB0[4],"/",hex2 DB0[5],"20",hex2 DB0[6]

```

```

if DB0[3]==6 then LCDOUT$fe,1,"Saterday:",hex2 DB0[4],"/",hex2 DB0[5],"20",hex2 DB0[6]
if DB0[3]==7 then LCDOUT$fe,1,"Sunday",hex2 DB0[4],"/",hex2 DB0[5],"20",hex2 DB0[6]
return

getkey:
    Pause 50  ' Debounce

getkeyu:
    ' Wait for all keys up

    PORTB = 0  ' All output pins low

    TRISB = $f0  ' Bottom 4 pins out, top 4 pins in

    If ((PORTB >> 4) != $f) Then getkeyu  ' If any keys down, loop

    Pause 50  ' Debounce

getkeyp:
    ' Wait for keypress

    For col = 0 to 3  ' 4 columns in keypad

        PORTB = 0  ' All output pins low

        TRISB = (dcd col) ^ $ff ' Set one column pin to output

        row = PORTB >> 4  ' Read row

        If row != $f Then gotkey  ' If any keydown, exit

        Next col

    Gotogetkeyp  ' No keys down, go look again

    Return

gotkey:
    key = (col * 4) + (ncd (row ^ $f))

    'if (Key==16) then keyName= "Enter"
    'if (Key==15) then keyName="F3"
    'if (Key==14) then keyName="F2"
    'if (Key==13) then keyName="F1"
    'if (Key==8) then keyName="CANCEL"
    'if (Key==12) then keyName="CANCEL"

```

```

if (Key==11) then keyName=9

if (Key==10) then keyName=6

if (Key==9) then keyName=3

if (Key==8) then keyName=0

if (Key==7) then keyName=8

if (Key==6) then keyName=5

if (Key==5) then keyName=2

if (Key==3) then keyName=7

if (Key==2) then keyName=4

if (Key==1) then keyName=1

```

Return

Option:

LCDOUT\$fe,1,"Options >>>>"

Pause 2000

RETURN

Adj :

Lcdout \$fe, 1,"Please Enter Time"

Lcdout \$fe, \$c0,"Format hh mm ss"

Gosubgetkey

if (KeyName<3) then h1= KeyName

Gosubgetkey

h2=KeyName

Lcdout \$fe, \$94, HEX h1, HEX h2

Gosubgetkey

if (KeyName<6) then m1= KeyName

Gosubgetkey

m2=KeyName

Lcdout \$fe, \$94, HEX h1,HEX h2, " ", HEX m1,HEX m2

Gosubgetkey

```

if (KeyName<6 ) then s1= KeyName

Gosubgetkey

s2=KeyName

Lcdout $fe, $94, HEX h1,HEX h2, " ",HEX m1,HEX m2, " ", HEX s1,HEX s2

pause 10

Lcdout $fe, $D4,"To Set Press Enter"    '1

Gosubgetkey

if (Key==16) Then    'enter

s= ((s1*16)+s2) : m= ((m1*16)+m2) : h= ((h1*16)+h2)

I2CWRITE SDA,SCL,$D0,$00,[s, m, h, DB0[3],DB0[4],DB0[5],DB0[6],$90]

endif

pause 50

I2CREAD SDA,SCL,$D1,$00,[STR DB0\8]

pause 10

Lcdout $fe,1," Time ",hex2 DB0[2],":",hex2 DB0[1],":",hex2 DB0[0]

Pause 3000

return

'=====

cont1:

I2CREAD SDA,SCL,$D1,$00,[STR DB0\8]

pause 100

Lcdout $fe, 1, hex2 DB0[2],":",hex2 DB0[1],":",hex2 DB0[0]

pause 1000

'Contvarbit[32]

For B0 = 1 To 32

Cont[B0]=0

Next B0

CHECK:

Lcdout $fe,$c0, "Start Monitoring"

```

```

if (PORTA.5=0) THEN  ' eeprom is in use.

PAUSE 200

GOTO CHECK

ENDIF

TRISA.5=0

PORTA.5= 0  ' Im using the eeprom

For B0 = 0 To 7

I2CREAD SGA,SGL,%10101011,B0,[BUSSS[b0]]  ' Read

pause 20

Next B0

IF BUSSS[0]>BOSSS[0] then high Soud

BOSSS[0] = BUSSS[0]

IF BUSSS[1]>BOSSS[1] then high Soud

BOSSS[1] = BUSSS[1]

IF BUSSS[2]>BOSSS[2] then high Soud

BOSSS[2] = BUSSS[2]

IF BUSSS[3]>BOSSS[3] then high Soud

BOSSS[3] = BUSSS[3]

Lcdout $fe,1,"2=",#BUSSS[2]

Lcdout $fe,$c0,"3=",#BUSSS[3]

pause 3000

PORTA.5=1  'm not using thrEEprom.

Con1= BUSSS[0]& %00000001:Con2=BUSSS[0]& %00000010:Con3=BUSSS[0] &
%00000100:Con4=BUSSS[0]& %00001000

Con5= BUSSS[0]& %00010000:Con6=BUSSS[0]& %00100000:Con7=BUSSS[0]&
%01000000:Con8=BUSSS[0]& %10000000

Con9=BUSSS[1]& %00000001:Con10=BUSSS[1]& %00000010:Con11=BUSSS[1]&
%00000100:Con12=BUSSS[1]& %00001000

Con13=BUSSS[1]& %00010000:Con14=BUSSS[1]& %00100000:Con15=BUSSS[1]&
%01000000:Con16=BUSSS[1]& %10000000

```

```

Lcdout $fe, 1 ' Clear screen

if Con1>0 then Lcdout $fe,1,"1-E.Safty Sys" : Cont[1]=1

if Con2>0 then Lcdout $fe,$c0,"2-E.Over Speed" : Cont[2]=1

if Con3>0 then Lcdout $fe,$94,"3-E.Fire Alarm" : Cont[3]=1

if Con4>0 then Lcdout $fe,$D4,"4-G1 Pre Alarm" : Cont[4]=1

pause 3000

Lcdout $fe, 1 ' Clear screen

if Con5>0 then Lcdout $fe,1,"5-G1 Stop Alarm" : Cont[5]=1

if Con6>0 then Lcdout $fe,$c0,"6-G1Temp.High" : Cont[6]=1

if Con7>0 then Lcdout $fe,$94,"7-G2 Pre Alarm" : Cont[7]=1

if Con8>0 then Lcdout $fe,$D4,"8-G2 Stop Alarm" : Cont[8]=1

pause 3000

lowSoud

Lcdout $fe, 1 ' Clear screen

if Con9>0 then Lcdout $fe,1,"9-G2 Temp high" : Cont[9]=1

if Con10>0 then Lcdout $fe,$c0,"10-Lub Oil Seper." : Cont[10]=1

if Con11>0 then Lcdout $fe,$94,"11-Oily Water Seper." : Cont[11]=1

if Con12>0 then Lcdout $fe,$D4,"12-Stear Gear1" : Cont[12]=1

pause 3000

Lcdout $fe, 1 ' Clear screen

if Con13>0 then Lcdout $fe,1,"13-Fire Detection" : Cont[13]=1

if Con14>0 then Lcdout $fe,$c0,"14-Smoke Detection" : Cont[14]=1

if Con15>0 then Lcdout $fe,$94,"15-AC Fail" : Cont[15]=1

if Con16>0 then Lcdout $fe,$D4,"16-Boiler 2 Fail" : Cont[16]=1

pause 3000

LOW denab

Shiftout Dout, Cout, 1, [%00000000\8,%0011111\8,%00000000\8,%00000000\8,_
%0011111\8,%00000000\8,%00000000\8,%0011111\8]

high denab

```

Pause 20

LOW denab

Shiftout Dout, Cout, 1, [%00000000\8,%1000\4,Cont[16]\1,%0\1,Cont[15]\1,Cont[14]\1,Cont[13]\1,%0\1,
Cont[12]\1,_Cont[11]\1,Cont[10]\1,%0\1,Cont[9]\1,Cont[8]\1,Cont[7]\1,%0\1, Cont[6]\1,
Cont[5]\1,Cont[4]\1,Cont[3]\1,Cont[2]\1,Cont[1]\1,_ %1000\4, %0000\4,%00000000\8,
%00000000\8,%1000\4,%0000\4,%00000000\8,%00000000\8]

high denab '

goto Cont1

Return

Loop0:

highSoud

Lcdout \$fe, 1 ' Clear screen

Pause 100

Lcdout \$fe,\$c0, "--- Testing ----"

Pause 2000

lowSoud

LOW denab

ShiftoutDout, Cout, 1, [%00000000\8,%0011111\8,%00000000\8,%00000000\8,_
%00111111\8,%00000000\8,%00000000\8,%00111111\8]

high denab

Pause 20

LOW denab

ShiftoutDout, Cout, 1, [%00000000\8,%1000\4,%1010\4,Pattern1\8,Pattern1\8,_
%1000\4,%1010\4,Pattern1\8,Pattern1\8,%1000\4,%1010\4,Pattern1\8,Pattern1\8]

high denab '

pause 1000

highSoud

LOW denab

ShiftoutDout, Cout, 1, [%00000000\8,%00111111\8,%00000000\8,%00000000\8,_
%00111111\8,%00000000\8,%00000000\8,%00111111\8]

high denab

```

Pause 20

LOW denab

Shiftout Dout, Cout, 1, [%00000000\8,%1000\4,%0101\4,Pattern2\8,Pattern2\8,_
%1000\4,%0101\4,Pattern2\8,Pattern2\8,%1000\4,%0101\4,Pattern2\8,Pattern2\8]

high denab

pause 1000

lowSoud

Bey =Bey +1

if (Bey<5) Then

  goto Loop0

endif

if (Bey=5) Then

  Pattern1= %00000000 : Pattern2= %00000000

  pause 1000

  goto Loop0

endif

return

end

```

Appendix B

'PicBasic Pro program to MCU2'ver 2.9

```

Define LOADER_USED 1

include "modedefs.bas"

Define OSC 20

Buusvar PORTD

BUSSS var byte[8]

Counvar byte

SGL var PORTC.0 'Clock pin

SGA var PORTC.1 'Data pin

B0 var byte 'Address

```

```
TRISA=0

loop1:
PORTA.0=0
PORTA.0=1 'freeze data
pause 10
PORTA = %00011111 '////1////OE1=0
pause 10
BUSSS[0]= Buus 'Board #1
PORTA = %00011101 '////2////OE1=0
pause 10
BUSSS[1]= Buus 'Board #2
pause 50
'=====
PORTC.4=1 'reset
pause 20
PORTC.4=0 'start
pause 50
PORTA=%10011000 'Bus=z,ALE,
PORTC.5=0
PORTC.6=0
PORTC.7=0
PORTA.0=0
BUSSS[2]= Buus
PORTA.0=1
PORTA=%10011000 'Bus=z,ALE,
PORTC.5=0
PORTC.6=0
PORTC.7=1
PORTA.0=0
```

```
BUSSS[3]= Buus
PORTA.0=1
BUSSS[4]= 0
BUSSS[5]= 0
BUSSS[6]= 0
BUSSS[7]= 0
TRISB.7=1
CHECK:
if (PORTB.7=0) THEN 'if eepromin use.
PAUSE 200
GOTO CHECK
ENDIF
TRISB.7 = 0
PORTB.7 = 0  'I'm reading
For B0 = 0 To 7
I2CWRITE SGA,SGL,%10101010,B0,[BUSSS[B0]]  ' Write each location
Pause 20
Next B0
PORTB.7 = 1  'I'm not reading
TRISB.7=1
pause 1000
goto Loop1
end
```

